Λύσεις για τις Κάποιες από τις Ασκήσεις του Φυλλαδίου

2. Εύρεση λαθών

Να βρεθούν τα λάθη στις παρακάτω περιπτώσεις:

Type mismatch

```
fun inc x = x + 1
fun f n = inc true
```

myprog.sml:2.11-2.18 Error: operator and operand don't agree (tycon mismatch)
operator domain: int
operand: bool
in expression:
inc true

Unresolved overloading

```
fun divide_by_two x = x / 2
```

myprog.sml:1.18 Error: overloaded variable not defined at type
symbol: /
type: int

Σωστότερα:

```
fun divide_by_two x = x / 2.0
```

Syntax errors

```
datatype 'a btree = Leaf of 'a
                  | Node of 'a btree * 'a btree
fun preorder Leaf(v) = [v]
  | preorder Node(l,r) = preorder l @ preorder r
```

myprog.sml:4.5-5.48 Error: data constructor Leaf used without argument in pattern
myprog.sml:4.5-5.48 Error: data constructor Node used without argument in pattern
myprog.sml:4.1-5.48 Error: pattern and expression in val rec dec don't agree (tycon mismatch)

*pattern: 'Z -> ('Z * 'Z) list*
*expression: 'Z -> 'Z * 'Z -> ('Z * 'Z) list*
in declaration:
preorder = (fn arg => (fn <pat> => <exp>))

Σωστότερα:

```
fun preorder (Leaf v) = [v]
  | preorder (Node(l,r)) = preorder l @ preorder r

datatype 'a option = NONE | SOME of 'a
fun filter pred l =
      let fun filterP (x::r, l) =
                case (pred x) of
                   SOME y => filterP(r, y::l)
                 | NONE => filterP(r, l)
            | filterP ([], l) = rev l
      in
        filterP (l, [])
      end
```

myprog.sml:8.23-8.28 Error: syntax error: deleting EQUALOP ID
myprog.sml:9.3-9.13 Error: syntax error: deleting IN ID

Σωστότερα:

```
fun filter pred l =
```

```
      let fun filterP (x::r, l) =
                (case (pred x) of
                    SOME y => filterP(r, y::l)
                  | NONE => filterP(r, l))
            | filterP ([], l) = rev l
      in
        filterP (l, [])
      end
```

Other errors

```
- let val e = 5 and f = e+1 in e+f end;
```
stdIn:24.23 Error: unbound variable or constructor: e

```
- val (x,x) = (2,3);
```
stdIn:22.1-22.18 Error: duplicate variable in pattern(s): x

```
- fun
=   f (x,y)   = x+y+1   |
=   f (x,y,z) = x+y+z+2 |
=   f _       = 3       ;
```
stdIn:39.1-42.15 Error: parameter or result constraints of clauses don't agree [tycon mismatch]

*this clause: 'Z * 'Y * 'X -> 'W*
*previous clauses: 'V * 'U -> 'W*
in declaration:
f =
(fn (x,y) => <exp> + <exp> + 1
| (x,y,z) => <exp> + <exp> + 2
| _ => 3)

```
- fun f(_) = "nonzero" | f(0) = "zero";
```
stdIn:43.1-43.37 Error: match redundant
_ =>
--> 0 =>

```
- fun h (head::rest) = head+1;
```
stdIn:1.1-2.23 Warning: match nonexhaustive
head :: rest => ...

```
- val succ = op + 1;
```
stdIn:54.1-54.18 Error: operator and operand don't agree [literal]
*operator domain: 'Z * 'Z*
operand: int
in expression:
+ 1

Σωστότερα:

```
- val succ = (fn x => op + (1,x));
val succ = fn : int -> int
```

4. Τέλειοι αριθμοί

Ένας «τέλειος» αριθμός ισούται με το άθροισμα των παραγόντων του, συμπεριλαμβανομένης της μονάδας (1) αλλά μη συμπεριλαμβανομένου του εαυτού του. Για παράδειγμα, το 6 είναι ένας τέλειος αριθμός γιατί $6=3+2+1$. Να γράψετε μία συνάρτηση που να ελέγχει αν ένας αριθμός είναι τέλειος.

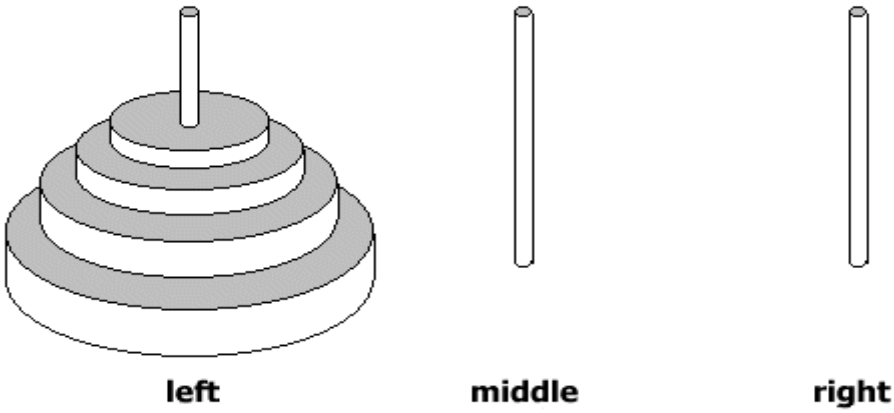
```
fun is_perfect n =
  let fun add_factors 1 = 1
        | add_factors m =
            if n mod m = 0
            then m + add_factors (m-1)
            else add_factors (m-1)
  in
    (n < 2) orelse (add_factors (n div 2) = n)
  end
```

6. Δυναμοσύνολο

Να ορίσετε μία συνάρτηση στην οποία όταν δίνεται ένα σύνολο με τη μορφή λίστας να επιστρέφει το σύνολο των υποσυνόλων του.

```
fun powerset [] = [[]]
  | powerset (h::t) =
      let
          fun cons h t = h :: t
          val pst = powerset t
      in
          (map (cons h) pst) @ pst
      end
```

8. Οι πύργοι του Hanoi



Υποθέστε ότι σας δίνονται τρεις ράβδοι και n δίσκοι διαφορετικού μεγέθους. Οι δίσκοι μπορούν να στοιβαχθούν στις ράβδους σχηματίζοντας πύργους. Βρίσκονται αρχικά στη ράβδο left με φθίνουσα τάξη μεγέθους. Πρέπει να μεταφερθούν στη ράβδο right ώστε τελικά να βρεθούν τοποθετημένοι με την ίδια σειρά. Αυτό πρέπει να επιτευχθεί με τους παρακάτω περιορισμούς:

- Σε κάθε βήμα μόνο ένας δίσκος μεταφέρεται από μία ράβδο σε άλλη.
- Ένας δίσκος απαγορεύεται να τοποθετηθεί πάνω από μικρότερο δίσκο.
- Η ράβδος middle μπορεί να χρησιμοποιηθεί για προσωρινή τοποθέτηση των δίσκων.

Να γράψετε μία συνάρτηση που να επιτυγχάνει το επιθυμητό αποτέλεσμα.

```
fun list_to_string str lst =
  let
    fun aux [] = "]"
      | aux [x] = (str x) ^ "]"
      | aux (x::xs) = (str x) ^ ", " ^ (aux xs)
  in
    "[" ^ (aux lst)
  end

fun hanoi_print a b c =
  let
    val str = list_to_string Int.toString
  in
    map print [str a, " - ", str b, " - ", str c, "\n"]
  end

fun hanoi a b c =
  let
    fun move 1 (x::xs) b c = (xs, x::b, c)
      | move n a b c =
          let
            val (a, c, b) = move (n-1) a c b
            val (a, b, c) = move 1 a b c
            val (c, b, a) = move (n-1) c b a
          in
            (a, b, c)
          end
  in
    move (length a) a b c
  end
```